福井大学
工学部　研究報告
第37巻　第2号
1989年9月

Wnnによる日本語入力の操作性を改善するためのプログラムpwnn

川口　湊*　鈴木重治*　寺島賢哉*　篠　競*

Pwnn: a Program for Improving Japanese Language Input Operation under wnn

Minato KAWAGUTI*, Shigeharu SUZUKI*, Ken-ya TERASHIMA* and Kisou SHINO*

(Received Aug. 30, 1989)

概要

'Pwnn' is a supplementary program of compact size used in conjunction with wnn and Njove. It is pivotal in improving wnn's critical human interface needed by Njove for editing Japanese language texts. When it recognizes command sequences Njove uses, it suppresses wnn from evaluating them erroneously, and lets these character strings through wnn intact. It also offers excellent modifiable key arrangement for technical touch typists, including scientists and academic authors, by dynamically reassigning various control keys.

1　序論

日本語エディタであるNjove [1] [2] に対して合理的な入力方法を実現することを意図して日本語入力用フロントエンドプロセッサpwnnを作成した。

そもそもNjoveはEmacs型の英文エディタとして使い易さの面で既に定評のあるJOVE [3] を日本語でも使えるように拡張改良したものであり、日本文字と英字や数式などが混在する文書の効率的な作成および編集に適する。Njoveでは、漢字の入力機構(かな漢字変換機構)はエディタ本体には含まず、かな漢字変換フロントエンドプロセッサとして外部に存在するものを使用する。

少なくとも、wnn [4] [5] [6] [7] はUNIXの標準的なかな漢字変換フロントエンドプロセッサとしてすべてのUNIXシステムで利用することが可能になると思われる。そこで、かな漢字変換機構をNjove本体から切り離した。Njoveはその分だけコンパクトになり、異なるUNIXシステムへの移植も容易になる。

一方で、エディタではそのヒューマンインターフェイスの違いが使い易さを大きく左右することから、好まれるかな漢字変換の方法は使用者毎に著しく異なる可能性がある。従って、Njoveではかな漢字変換の方式の選択を使用者の自由な裁量に委ねる。

言い換えれば、Njoveが実際に使い易い日本語用エディタとして機能するには二つの要件を同時に満足する必要がある。第一に、Njoveそれ自体が日本語の編集に真に適するように設計されていなければならない。第二に、Njoveに対する日本語入力のフロントエンドとなる部分の性能をこれと平行して向上させることも重要である。

*情報工学科 Department of Information Science

一般的な状況下では、Njove はそのかな漢字変換に wnn が採用されることを想定しているので、Njove の操作性の少なからぬ部分は現実には wnn の操作性に強く拘束される。wnn 自体のヒューマンインターフェイスの機能 (界面機能) は、Emacs 型のエディタの持つコマンド体系を強く意識して作られてはいるが、それでもいくつかの改善すべき点がある。そこで、Njove と wnn とを組み合わせて日本語の文書を編集する際の作業効率を向上させる新たな工夫が必要になる。そのために、ターミナルと wnn の間に、wnn に対するフロントエンドプロセッサとしての役割を果たす pwnn を新たに挿入する。pwnn はターミナルに最も近い位置を占めて、使用者が入力するターミナルからの生のデータに最初に対応し、使用者の意図に従ってそれらのデータを指定された方法で処理してその結果を後続の wnn に渡す。そこで wnn は要求に応じてローマ字をかなに変換し、更に続けてかな漢字変換を行なう。このようにして、wnn の出力であるその変換結果がターミナルからの入力の本来の目的である '使用者のプログラム'[1]への入力となる。

wnn はそれ自体で完結した日本語入力用のフロントエンドプロセッサであるので、更にもう一段その wnn に対するフロントエンドプロセッサを重ねることになる。このように pwnn を wnn に重ねる方式には次の利点がある。

- wnn が利用できる UNIX システムで Njove の利点を簡便に生かすことができる。
- wnn の開発状況とはある程度独立に、必要な機能を備えた pwnn を作成できる。
- コンパクトな pwnn をそれぞれの使用者がカスタム化して、wnn 自体は '標準' のまま利用することが可能である。こうすると、例えば、各人が自分の pwnn を携行することによって、他のシステムで日頃の慣れた入力環境を再現するのが容易になる。

このような方針から、pwnn は wnn の配布されている版をそのままの形で利用する[2]。pwnn の採用によって、Njove での編集に必要な界面機能のかなりの部分は実用面でほぼ満足できるところまで実現できる。

以下の記述では、特に断らないかぎり pwnn に固有の部分と wnn に属する部分とを一体化して pwnn と呼ぶ。しかしながら、特に wnn それ自体の機能がそのままの形で表面に出ている部分についての記述である場合には、'pwnn' と 'wnn' とを区別する。

2 pwnn の概要

一つのターミナルから shell に対して pwnn を起動させると、pwnn は擬似端末 (pty) を獲得して更に wnn を起動する。従って、論理的には二つの特殊なプログラム (pwnn と wnn) とがカスケード状に通信回線の中途に擬似的に挿入された状態になる。この状態は、次にコマンドレベルで exit を入力して pwnn を終了させるまで持続する。このような状態では、あるプログラムがそのターミナルに対して入出力を行なうと、その入出力データはすべて途中で pwnn および wnn を経由するので、そのターミナルからのキーボード入力は、pwnn および wnn が生のデータを処理した後のものが、あたかもターミナルから直接入力されたかのようにそのプログラムに入力される。プログラムからの出力も同様に pwnn によって途中で処理されて物理的なターミナルに送られる。

pwnn が起動されている状態では、ターミナルの画面の最下段の 1 行[3]が wnn の専用領域として割り振られる。このあらかじめ取り置かれた領域を wnn の変換行と呼び、wnn がターミナル使用者と対話するときの窓として排他的に使用する。

---

[1] Njove がその代表例であるが、pwnn は shell のコマンドラインを含めたあらゆる状況の下で全く同様に使用できる。例えば、grep が日本語を受け入れるように拡張されていれば、その引数に日本語の文字列を用いることができる。以下ではこれらを総称して単に 'プログラム' と呼ぶことにする。

[2] pwnn は wnn の現行版である Wnn Version 3 に基づいて作成されているが、近く配布が始まる Version 4 においても wnn の基本的なインターフェイス仕様は変更されていないので、pwnn はそのままの形で Wnn Version 4 にも対応できると思われる。

[3] default では 1 行だが、オプション指定で任意の行数を変換行として使用できる。

pwnn の主な機能はターミナルからの日本語入力に際して、実時間で対話的に入力データのローマ字かな変換およびそれに続くかな漢字変換を行なうことである。そこで、pwnn には 4 種類のモードを持たせて、それぞれを異なる字種の出力に対応させる。これらのモードの間の遷移は、pwnn のそれぞれのモード毎に定義された‘制御キー’と呼ぶ 1 種類から 3 種類のキーの一つをキーボードから叩くことによってターミナル使用者が意のままに行うことができる。

pwnn の開発の最重点対象である Njove での日本語入力の最適化を図るために、Njove の特定のコマンドに対しては pwnn はそのモードを自動的に遷移したり、一時的に wnn の変換を抑制したるする機能も合わせ持つ。Njove のどのコマンドに対してどのようにモードを遷移させるかはすべて、個々の使用者が pwnn のカスタム化のためにあらかじめ作成する特別のファイルの中で指定できる[4]。

3 pwnn の設計の基本方針

pwnn の設計では、優れた性能の日本語端末装置を用いて touch typing で作業する場合の科学技術文書の入力時の作業効率を最優先するようにした。

3.1 ローマ字入力に対する最適化

ほとんどの日本語端末装置のキーボードは‘カナ’キーによって、ASCII コードと半角カタカナコードとを使い分けることができるように設計されている。従って直接半角カタカナコードを入力してそれを基にかな漢字変換を行なわせることも勿論可能であるが、現実には UNIX による科学技術文書の作成にはほとんどの場合ローマ字入力方式が採用されているようである。ローマ字入力が好まれる理由は概ね次のようなものであると考えられる。

- 当然のことながら、UNIX では ASCII コードを常用するから ANSI キー配列に慣れている。JIS カナキー配列を混用すると混乱を生じて能率が逆に低下する。
- JIS カナキー配列は 3 段ではなく 4 段のキー配列を用いて文字を入力しなければならないので、touch typing がやりにくい [8]。
- JIS カナキー配列ではカナ入力においても shift キーを常用するが、touch typing の際にはこれは速度を低下させ誤打を増加させる原因の一つでもある [8] [9]。

そこで、pwnn ではローマ字入力に対して最適化を行なう。更に、pwnn では touch typing 時に偶然‘カナ’キーに指が触れて不本意に半角カナコードが入力されるのを未然に防ぐこともできる。ソフトウェアによって実現されているこの機構を pwnn では‘カナキーロック’機構と呼ぶ。

3.2 使用する文字コードの選択

pwnn は、Njove において採用している字種を合理的に処理できるように特別の配慮をする。そのために、次のような条件に適合するように使用する字種を定める。

3.2.1 使用する日本文字コードは拡張 UNIX コードに限る。

UNIX システム内部の不要な混乱を回避するためにも、すべての UNIX システムが一つの合理的な日本語の文字コード体系に統一されるのが望ましい。そこで、Njove では日本文字コードとして一貫して拡張 UNIX コード (EUC)

[4] pwnn と Njove とが連携して機能する極めて限られた一部のコマンドを別とすれば、この機能は対象を Njove に限定するものではないから、Nemacs に対してもまったく同様の pwnn のカスタム化が可能である。これは、これらの二つのエディタのコマンド体系がほぼ共通したものであることに基づいている。

を用いる。従って、pwnn もまたひらがな、カタカナ、漢字などの日本文字に対して 2 バイトの EUC コードを用いる。

3.2.2 使用する文字セットの全体に重複がないようにする。

pwnn でも Njove と同じように

1. 半角のカナ

2. 全角の記号、数字、アルファベット、空白文字などのような、ASCII 文字と重複する文字

は使わない。基本的には同じ文字を指す 2 種類のコードを併存させることによって生じる不必要な混乱は日本語の文書処理全般にわたる障害となる可能性がある。例えば、grep による文字列の検索や、TEX による清書 (製版) 処理の際に、人間の思考の上では同じ文字として扱われているものに、外見が酷似した '2 つの文字' として別のコードを割り振るのは不条理であり、実際面でも不都合が生じる。

Njove では、今後の文書処理の趨勢として、日本語の文書についても出力の美的印象が問題となる文書は然るべき専門の清書プログラム[5]を日常的に使用するようになることを想定している。従って、出力時における異なる大きさの字体の選択は入力時に同一文字に対して複数種類存在するコードの一つを指定するのではなく、独立した別個の清書プログラムがその選択を担当することを暗黙のうちに仮定している。

このような方針に沿って、使用者が意図的に指示しない限り、pwnn は入力時における半角カナおよび全角記号などの混入を積極的に排除する。

3.3 UNIX システムでの日本語文書処理における touch typing 入力への配慮

UNIX システム全般で広く使用されている標準的な ANSI キーボードを用いてローマ字による日本語文書の作成や編集を行なう際に、英文における作業と同様に touch typing を採用して効率を上げ、あわせて、入力作業の疲労を極力軽減したい。そのために、ローマ字入力を採用することは既に述べた。

ところで、特に科学技術系の文書においては、キーボードのホームキーポジションから指を離さずに、ASCII 文字、ひらがな、カタカナ、漢字を随時切り替えながら入力できるのでない限り、日本語入力を touch typing で行なうのは現実的でない。

そこで、実用に耐える日本語のための touch typing を実現するために、pwnn には幾つかの独自の工夫を施した。後で述べるアスキーモードの導入、制御キーによるモードの切り替え、キーを修飾するという方法はいずれも touch typing での効率を向上させるためのものである。

3.4 使用者の多様な使用形態に適応できる柔軟性

pwnn が具備している独自の機能の多くは、その一つひとつを使用するか否かを pwnn の起動時のオプション指定で選択できる。

更に、Njove のコマンドの一つひとつに対する pwnn の対応についての指定を特別のファイルに書き込むことによって pwnn のカスタム化が可能である。

これ以外にも、キーの配置を自由に設定することができる[6]。例えば、UNIX の制御には qwerty キー配列、英文入力には Dvorak キー配列、ローマ字入力には使用者が独自に考案したローマ字専用の特殊配列、の 3 者をそれぞれ同時に使用することも可能である。

---

[5]代表例は日本語化された TEX である。

[6]本稿ではキー配列の変更方法や制御キーならびに修飾キーを指定する方法に関しての詳細は割愛する。

4 pwnnの起動と終了

ターミナルから

```
% pwnn -i -s -q
```

と入力すると[7]、

```
Wnn(かな漢字変換フロントエンドプロセッサ)
%


[---]
```

というwnnのメッセージが画面に出て、pwnnが起動されたことを示す。これ以降はターミナルの入出力はすべてpwnnを経由することになる。

pwnnを終了するには、後述の透過モードまたはアスキーモードのときにshellに対して

```
% exit
```

と入力すればよい。

5 入力字種の選択のためのpwnnの4種類のモード

pwnnは常に

1. 透過モード
2. ひらがな変換モード
3. カタカナ変換モード
4. アスキーモード

の4種類のモードのいずれかにあり、それぞれのモード毎に入力する文字列をどのように変換するかが異なる。図1に示すように、3種類のキー(主制御キー、副制御キー、補助制御キー)によってモードの切り替えを行う[8]。

5.1 透過モード

pwnnが起動した直後の段階ではキーボードからの入力は、Back Space以外は、すべて叩いたキーの表面に表示してあるとおりのASCIIコードそのものである[9]。

このとき画面の最下行である変換行の左端には、wnnが'[---]'というモード表示マークを表示し、pwnnが'透過モード'にあることを示す[10]。

pwnnでは主制御キー[11]としてBack Spaceキーを用いる。主制御キーを押す毎に透過モードと'ひらがな変換モード'とが交互に入れ替わる。UNIXにおいて必要とされるすべてのキー入力はpwnnの透過モードの下で行うことができるから、日本語入力の有無にかかわらずpwnnを起動しておいて透過モードで使用し、日本語入力の必要が生じた時に変換モードに切り替えて入力すると円滑に日本語入力が実行できる。

---

[7] 後で述べるオプションは -i -s -q という形で指定する。任意の数と組み合わせを選ぶことができる。

[8] pwnnでは使用者がこれらの制御キーの配置位置を自由に設定できるが、以下ではdefaultの設定を採用したものとして記述する。

[9] pwnnの初期状態のdefault値は透過モードに設定されている。

[10] 実際には後述の'アスキーモード'でも同一のマークが左下に表示される。

[11] UNIXのコマンドの入力においても、wnnによるローマ字かな変換においても、あるいはNjoveでの日本語文書編集においても、通常Back Spaceキーは使用する必要がない。そこで、pwnnではこのキーを主制御キーとして使用する。

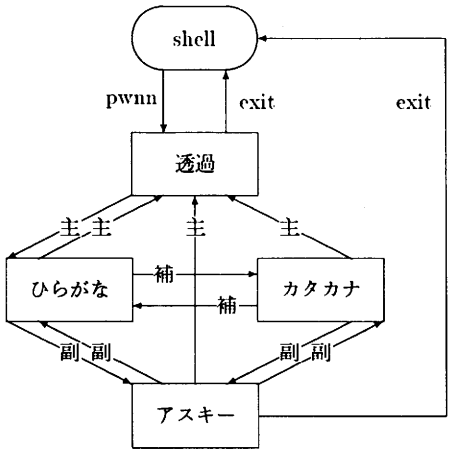

図 1: 3 種類の制御キーによるモード間の遷移

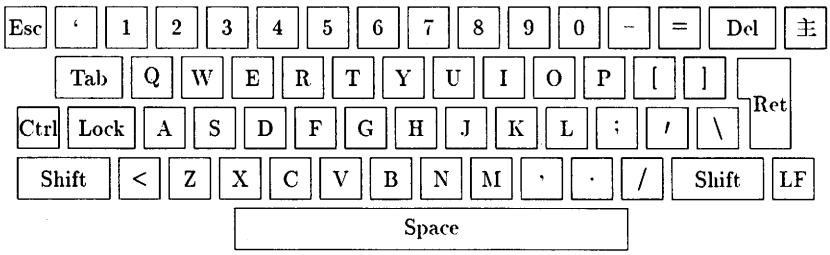

図 2: 透過モードにおける設定

5.2 ひらがな変換モード

pwnn が透過モードにあるとき、主制御キーを一度押すとひらがな変換モードになる。このモードではキーボードからの入力はローマ字入力と解釈されて、wnn のローマ字かな変換機構がひらがなに逐次変換する。wnn はひらがなの文字列の部分に対してのみかな漢字変換を行うから、日本語の入力作業では専らこのひらがな変換モードを利用することになる。変換行の左端には '[あ r]' というモード表示マークが表示される[12]。

5.3 カタカナ変換モード

ローマ字入力をひらがなではなく、カタカナに変換したい場合には、補助制御キーによって pwnn のモードをカタカナ変換モードに変えればよい。補助制御キーとして pwnn では L キーを用いる[13]。

補助制御キーを一回押す毎に pwnn のモードはひらがな変換モードとカタカナ変換モードとの間を交互に遷移す

[12] 変換行は常に wnn が直接管理しているから、前の '[---]' と同様に、これも wnn が出す表示そのものである。

[13] ローマ字入力では通常 l(ℓ) を使用する必然性はないので、pwnn の標準のキーの設定では補助制御キーを L キーに設定して使用するように設計した。

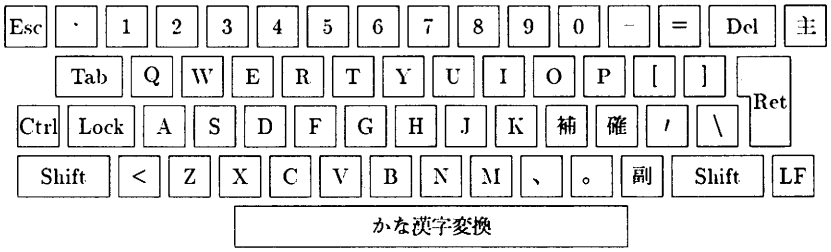

図 3: かな変換モードにおける設定

る。

5.4 アスキーモード

ローマ字かな変換による日本語入力の途中で ASCII 文字を入力するには一時的にローマ字かな変換モードから抜け出て無変換で ASCII コードを入力する機能が必要である[14]。

そこで、pwnn では透過モードとは別個に ‘アスキーモード’ を設け、かな変換モードとアスキーモードとの間の切り替えを副制御キーによって行う。副制御キーとして pwnn では / キーを用いる[15]。

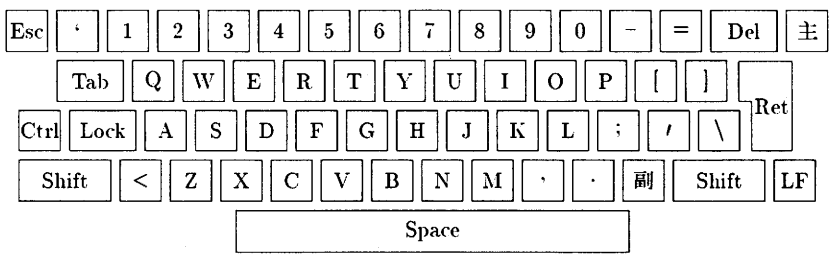

図 4: アスキーモードにおける設定

主制御キーを用いて透過モードにすれば ASCII コードを入力できるが、この方法は日本文字と英字の両文字が混在する文書の編集にはかならずしも能率のよい方法ではない。透過モードでは主制御キーを除く全てのキー入力をそのまま pwnn ないしは wnn を透過させ、wnn を経由しないで直接入力する場合とほぼ同じ使い勝手を実現する点に最大の主眼がある。そうすると、主制御キーとしてモードの切り替えに使用できるキーは ‘文字’ として使用される可能性が最も少ないキーでなくてはならない。

[14] pwnn は透過モードでは ASCII コードを出力するが、このときキーボードの入力を pwnn で加工せずにそのまま透過させることを保証している。これに対しアスキーモードでは叩かれたキーと出力する ASCII コードとの関係はそれぞれの利用者が自由に設定できるように設計されている。仮に、大胆なキーの再配置を行ったためにアスキーモードで使用者が混乱に遭遇したとしても、主制御キーによってアスキーモードは必ず透過モードに遷移できる。従って、透過モードの標準のキー配置に戻ることにより容易にターミナルの制御を回復できる。

[15] ローマ字入力では使用する必要がないが touch typing において楽に叩くことができるキーとして / キーを採用した。ちなみに、/ キーの近くの ’ キーはローマ字入力の際にはナ行の入力と ‘ん’ の入力の区別をするのに使用する。従って、他の目的、例えば副制御キーとして用いることはできない。

一方で、アスキーモードではtouch typingに際してASCIIコードを入力するのが主な狙いである。透過モードとは異なり、使用頻度の低い記号キーを含めたキーボード上の全てのキー入力をASCIIコードに対応するコードとして受けつけなければならないという制約はアスキーモードにはない。既に述べたように、UNIX全般の入力作業は透過モードのもとで支障なく行なうことができるから、これと相補的なアスキーモードは専ら日英の文字列が混在した文書のASCII文字部分を能率よく入力するための特別のモードであると割り切って差し支えない。そこで、アスキーモードに対しては、そのような文書の入力の際にtouch typingでの高速入力を可能ならしめるようなキー設定を選択することができる。

特に科学技術系の文書では、日本文の途中に専門用語や数式記号などASCII文字が頻繁に使われる。そこで、日本文字と英字とを切り替える副制御キーを使用する機会は、記号としての/の出現頻度よりも格段に高いと考えられることから、pwnnでは副制御キーとして/キーを用いることとした。

5.5 モードの切り替えに伴うトラブルの回避

特にtouch typingで高速に入力を行う際に、頻繁にモードを切り替えることに付随して発生する入力ミスを極力減らす工夫をした。

キーボードのクリック音の周波数をソフトウェアで選択できる端末装置[16]では、pwnnはこの4種類のモード毎に明確に異なるピッチのクリック音を出す。これによって、画面を見ずに入力している時でも音の高さによって現在のモードを把握することができる。

wnnとは異なり、pwnnはかな変換モードから透過モードまたはアスキーモードに遷移する際に、変換行のデータを自動的にプログラムに排出する。これは、wnnの使用経験に基づく有効なトラブル回避策の一つである。

6 pwnnのオプションの選択

pwnnはその起動時に行うオプション指定のパラメータの選択によって、キーボード上のいくつかのキーに対して特別の機能を持たせることができる。このようにあるキーに別の意味を持たせることを、そのキーを‘修飾する’という。

6.1 -iオプション

ANSIキーボードを用いてtouch typingで入力を行う際に威力を発揮する。日本文字と英字とが混在する文書の作成には特に適している。このオプションを指定すると

1. かな変換モードにおいて次のキーが修飾される。

/	副制御キー(かな変換モードとアスキーモードとを入れ替えるキー)になる。
Space	かな漢字変換キーになる。
;	確定キーになる[17]。変換行の文字列をプログラムに排出して変換行を空にする。
l(ℓ)	補助制御キー(ひらがな変換モードとカタカナ変換モードとの間を切り替えるキー)になる。
\	\キーは特別の場合に‘無修飾化キー’として機能するように修飾される。無修飾化キーは、後続のキーが指定された修飾キーであった場合には\文字という意味を失い、直後のキーの修飾を一時的にキャンセルして本来のASCII文字のキーとして機能させる。それ以外の場合には通常の\

[16]例えば、ビクターデータシステムズ製のV4Wシリーズはこの機能を備えている。

[17]広く用いられている用語に準じて‘確定キー’と呼ぶが、機能を正しく表現するならば、wnnからデータを外部に排出させるという意味で、‘排出キー’と呼んだ方がよい。

キーとして機能する。\ キーが無修飾化するのは、Space、:、/ の各キーである。\ キーに続けてSpace、:、/ キーのいずれかを叩くことによって、-i オプションの下のかな変換モードにおいても、キー本来のASCII文字である空白、:、/ を入力することができる。言い換えれば、\ キーを叩くと、\ が表示されるが、続けて上記のキーの一つを叩くと\ がこれらの文字のひとつに自動的に置き換わる。このモードで文字としての\ の直後に修飾キーのひとつを使うには、\ の直後に C-g を叩く[18]。(C-g は\ の無修飾化機能を遮断する。)

2. アスキーモードでは

/ 副制御キー(かな変換モードとアスキーモードとを入れ替えるキー)になる。

\ \ キーの無修飾化の対象になるのは、/ キーだけである。\ キーの直後に / キーを叩くと、/ キーは -i オプションで述べた修飾されたキー(副制御キー)ではなく、キー本来のASCII文字(/)が入力される。\ キーを叩くと、\ が表示されるが、/ を入力すると\ が / に自動的に置き換わる。

このオプションを指定しないと、副制御キーも補助制御キーも定義されないから、アスキーモードへ遷移することはできない。従ってこのときは、アスキーモードは存在しないのに等しい。

6.2 -s オプション

このオプションを選ぶと、'標準設定'のwnnがかな変換モード時に実行する次のようなキーの'読み替え'をpwnnは抑止して、キーに記された本来のASCIIの記号を出力させる。特に科学技術系の文書の作成の際に標準的に使用するオプションである。

[wnnが'「'に変えるのを抑止して、'['を出力させる。

] wnnが'」'に変えるのを抑止して、']'を出力させる。

/ wnnが'・'(中点)に変えるのを抑止して、'/'を出力させる[19]。

6.3 -q オプション

このオプションでは、ローマ字入力の際に'な'行(および'ん')の入力のための'n'の働きはそのままにして、その他に、q キーを'ん'文字専用に割り当てる。伝統的なローマ字表記の一つの問題点は、'な'行と'ん'とに同じ文字'n'を割り振っているために、両者の識別が不必要にあいまいになっていることである。このオプションで導入した方法によって、入力のストローク数を増やさずに、'ん'に母音または'や'行が続くときの難点を回避することができる。計算機におけるローマ字入力は、外国人に正しい発音で読んでもらうことを狙いとしている訳ではなく、ひらがなとの一対一の対応関係のほうがはるかに重要である。従って、統一式(訓令式)表記法を採用して更にこの -q オプションを用いると入力の高速化の面では効果がある。

なお、wnnでは、かな変換モードの時にqキーを叩くと特殊な入力状態になる。このとき、モード表示マークは'[q]'に変わり、後続の入力はC-gが入力されるまですべてASCII文字のまま(ローマ字に変換されずに)変換行に入って行く。C-g によって元の変換モードに戻る。pwnn の -q オプションの下でwnnのこの機能を使用するには、-i オプションの場合と同じく\ キーで q キーを無修飾化すればよい。

---

[18] 一般に、コントロールキー(Ctrl キー)を押したまま x キーを押すことを C-x という形で表す。

[19] 中点(中黒)は\cdot を使って組み版するのが TEX の標準的な方法である。

6.4 -z オプション

pwnn によるカナキーに対するソフトウェアロックを外し、カナキーが使えるようにする。このオプションを指定しない default では、半角コードの入力を検出する毎に、pwnn は警告のベルを鳴らしてそのデータを破棄する。従って、カナキーに過失で触れて半角のカタカナコードが入力されたとしても、pwnn はその入力を確実に拒否する。

7 Njove のコマンドを wnn に捕捉させないための工夫

当初から Emacs を意識して設計されている wnn は、かな変換モードにおいて、C-x 型の制御コード単体のみよりなるコマンドを透過させるように作られている。しかしながら、Njove の C-x +‘文字列’型および ESC +‘文字列’型のコマンドでは C-x および ESC のみを透過させて、それに続く文字(列) を wnn は変換行に横取りしてしまい、Njove のコマンドの一部としてそのままの形でプログラムに排出してくれない。日本文の編集時に、かな漢字変換がいつでもできる状態を保ったままで(wnn の内部状態をあまり意識せずに) ポイントの移動や削除などの編集作業を行いたいが、これではそれができないので、Njove での作業効率を著しく劣化させる。このことは、日本文を英文と全く同等の使い勝手で編集したいという Njove の基本方針にそぐわない。

7.1 pwnn における解決方法

この問題は wnn のカスタム化機能によっても解決できない。そこで、‘wnn のフロントエンドプロセッサとしての pwnn’に動的に wnn を操作させることによって、(wnn を包含した意味での)pwnn は Njove 側の要求であるそのような機能を実質的に実現する。

具体的には、使用者のホームディレクトリに

```
.pwnn_term
```

というサブディレクトリを設け、その下に、使用するターミナルの種類毎に UNIX の環境変数 $TERM が識別するターミナル名をファイル名とするファイルを作る。このファイルには、wnn を透過させたいコマンドの文字列を例に示すような形式で記述しておく。ただし、透過させることができるコマンドは最初のコードが C-x で始まる文字列であるか、あるいは ESC コードで始まる文字列のいずれかでなければならない。

このように、使用者にファイルを作成させる方式を採用しているのは、そもそも Njove が ~/.joveterm/<terminal 名> によって自由なカスタム化を許しているからであり、~/.pwnn_term/<terminal 名> の内容もそれで定義されたものと同じ Njove のカスタム化されたコマンドの文字列を用いなければならない。

7.2 Njove のコマンド文字列を透過させるためのカスタム化ファイルの例

上で述べたカスタム化ファイルの一例を次に示す。

```
# Customization file for pwnn
# used with Njove
ESC { backward-paragraph
ESC C-B backward-s-expression
ESC A backward-sentence
ESC B backward-word
ESC < beginning-of-file
ESC , beginning-of-window
```

ESC C case-word-capitalize

ESC L case-word-lower

ESC U case-word-upper

ESC C-L clear-and-redraw

ESC W copy-region

C-X C-O delete-blank-lines

C-X K delete-buffer

C-X D delete-current-window

C-X 1 delete-other-windows

ESC \ delete-white-space

ESC > end-of-file

ESC . end-of-window

C-X E execute-keyboard-macro

C-X C-X exchange-point-and-mark

ESC J fill-paragraph

ESC M first-non-blank

ESC] forward-paragraph

ESC C-F forward-s-expression

ESC E forward-sentence

ESC F forward-word

C-X ^ grow-window

ESC D kill-next-word

ESC Rub kill-previous-word

C-X C-K kill-region

ESC C-K kill-s-expression

C-X Rub kill-to-beginning-of-sentence

ESC K kill-to-end-of-sentence

ESC ~ make-buffer-unmodified

C-X C-N next-error

C-X N next-window

ESC C-V page-next-window

C-X C-P previous-error

ESC V previous-page

C-X P previous-window

ESC Q query-replace-string

ESC Z scroll-down

C-X 2 split-current-window

C-X C-T transpose-lines

ESC Y yank-pop

pwnn はかな変換モードのときに C-x または ESC をキーボードから受け取ると、先ず自動的に透過モードに遷移し、後続の文字列を逐次透過させる一方で、それがこのファイルに大文字に変えて書かれている文字列のパターンのいずれかと一致するかどうか比較する。もし一致すれば、pwnn のモードを当初のモードに戻す。いずれとも一致しなければ、透過モードのままになっている。

ただし、一つだけ例外がある。'query-replace-string' については、ESC q と入力されたところで元のかな変換モードに戻るがそれに続く文字列の 2 度目の Return コードを受け取ると、再び透過モードに変わる。この対話的な置き換えコマンドは、置き換える対象となる文字列と、新しい文字列とを Return キーで区切りながら続けて入力する。この後で置き換えの指示を ASCII 文字の 1 文字を逐次入力することによって行う。従って、この最後の段階ではコマンドの一部として pwnn が ASCII 文字をそのまま排出しないと Njove は混乱してしまう。

8 結論

pwnn によって実現できた主要な機能は次の通りである。

1. 日本文字と英字とが混在するような文書を touch typing によって入力することが可能になった。
2. wnn のかな変換モード時でも Njove のコマンドを実質的に透過させることができるようになった。

この結果 wnn を用いて Njove で日本語の文書を編集する際の入力効率が大幅に向上した。特に英字を混ぜて使用する科学技術の分野の各種の文書や学術論文などの作成が容易になった。

参考文献

[1] 川口 湊, 鈴木 重治, 藤井 弘, 吉田 肇: Njove: 日本語機能を付加した UNIX 実時間表示エディタ JOVE, 福井大学工学部研究報告 **36** 263-272 (1988)

[2] 川口 湊, 鈴木 重治, 藤井 弘, 吉田 肇: JOVE の日本語化, jus 12th UNIX Symposium Proceedings pp.90-99 (1988)

[3] Jonathan Payne: JOVE Manual for UNIX Users, 4.3BSD 版マニュアル (1986)

[4] M. Hagiya, T. Hattori, A. Morishima, R. Nakajima, N. Niide, T. Sakuragawa, T. Suzuki, H. Tsuiki and T. Yuasa: Overview of GMW+Wnn system, Proc. 2nd IEEE Conference on Computer Workstations, Santa Clara, (1988)

[5] M. Hagiya, T. Hattori, A. Morishima, R. Nakajima, N. Niide, T. Sakuragawa, T. Suzuki, H. Tsuiki and T. Yuasa: Overview of GMW+Wnn system, Advances in Software Science and Technology **1**, (1989) to be published

[6] 桜川 貴司: 開かれた日本語入力システム Wnn, **bit 19** No.10, 1309-1319 (1987)

[7] 鈴木 隆, 立木 秀樹, 新出 尚之: Wnn 日本語入力システムの仮名漢字変換について, 日本ソフトウェア科学会第 4 回大会論文集, pp.99-102 (1987)

[8] 山田 尚勇: タイプライタとその入力の歴史的考察 2, **bit 13** No.7, 947-983 (1981)

[9] 山田 尚勇: タイプライタとその入力の歴史的考察 5, **bit 13** No.10, 1547-1556 (1981)